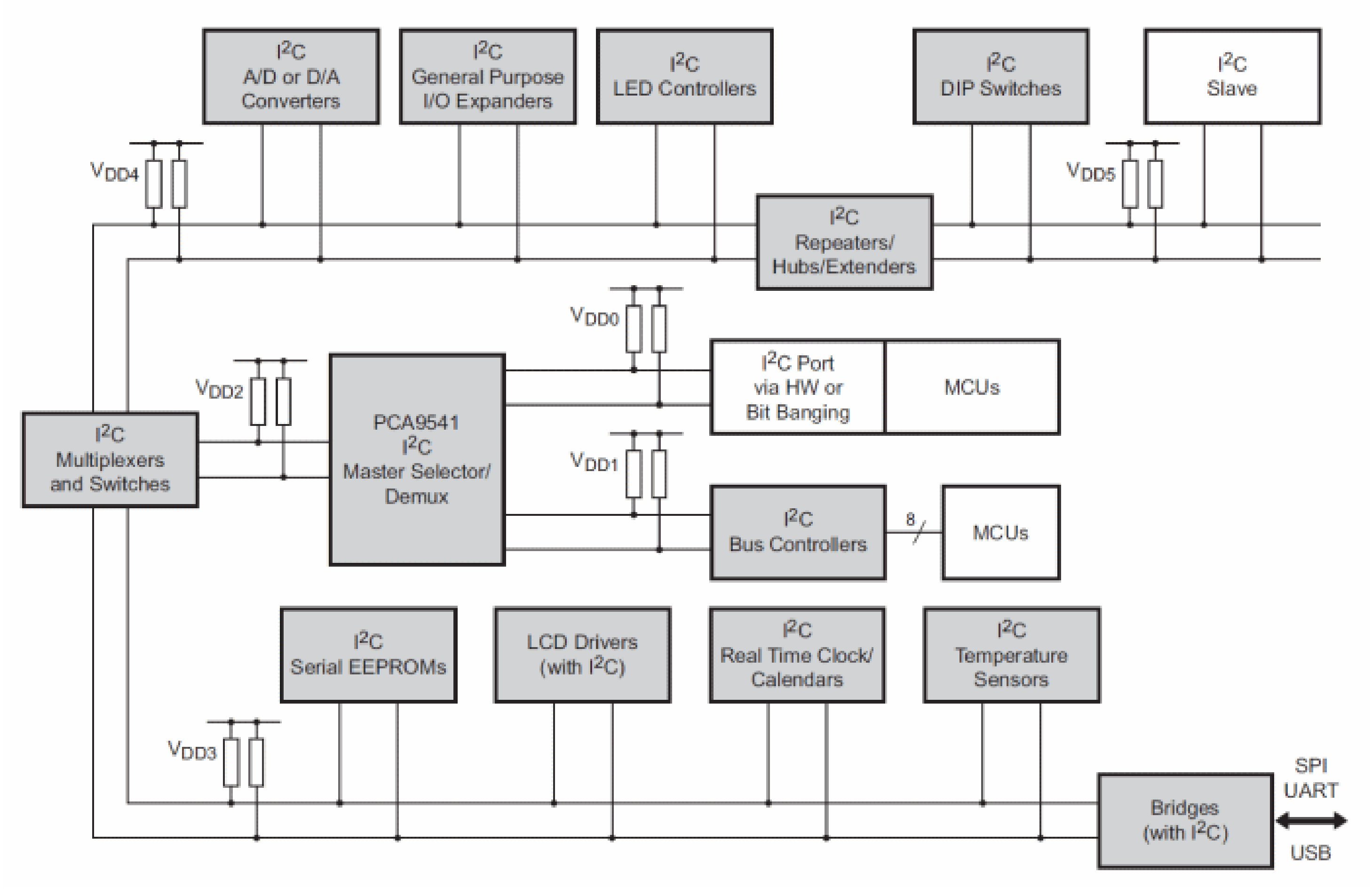
I2C Bus Specification

A typical embedded system consists of one or more microcontrollers and peripheral devices like memories, converters, I/O expanders, LCD drivers, sensors, matrix switches, etc. The complexity and the cost of connecting all those devices together must be kept to a minimum. The system must be designed in such a way that slower devices can communicate with the system without slowing down faster ones.

To satisfy these requirements a serial bus is needed. A bus means specification for the connections, protocol, formats, addresses and procedures that define the rules on the bus. This is exactly what I2C bus specifications define.


I2C A/D or D/A Converters
I2C General Purpose I/O Expanders
I2C LED Controllers
I2C DIP Switches
I2C Slave
VDD4
VDD5
I2C Repeaters/ Hubs/Extenders
VDD0
I2C Port via HW or Bit Banging
MCUs
VDD2
PCA9541 I2C Master Selector/ Demux
I2C Multiplexers and Switches
VDD1
I2C Bus Controllers
8
MCUs
I2C Serial EEPROMs
LCD Drivers (with I2C)
I2C Real Time Clock/ Calendars
I2C Temperature Sensors
VDD3
Bridges (with I2C)
SPI
UART
USB

The I2C bus uses two wires: serial data (SDA) and serial clock (SCL). All I2C master and slave devices are connected with only those two wires. Each device can be a transmitter, a receiver or both. Some devices are masters – they generate bus clock and initiate communication on the bus, other devices are slaves and respond to the commands on the bus. In order to communicate with specific device, each slave device must have an address which is unique on the bus. I2C master devices (usually microcontrollers) don't need an address since no other (slave) device sends commands to the master.

I2C terminology

Transmitter This is the device that transmits data to the bus

Receiver This is the device that receives data from the bus

Master This is the device that generates clock, starts communication, sends I2C commands and stops communication

Slave This is the device that listens to the bus and is addressed by the master

Multi-master I2C can have more than one master and each can send commands

Arbitration A process to determine which of the masters on the bus can use it when more masters need to use the bus

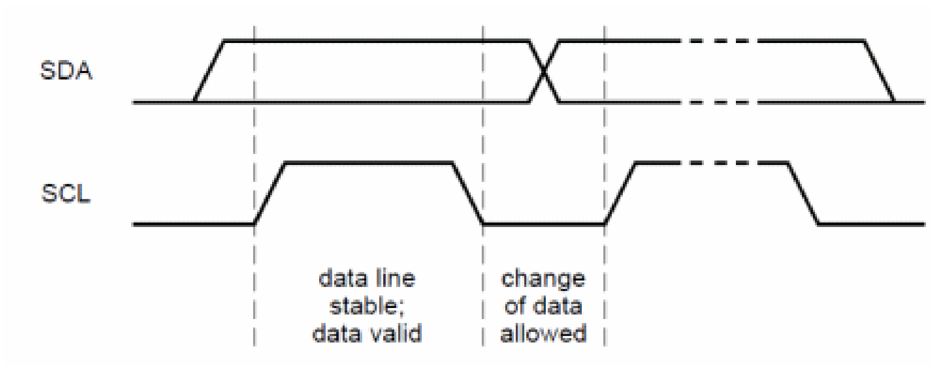
Synchronization A process to synchronize clocks of two or more devices

Bus Signals

Both signals (SCL and SDA) are bidirectional. They are connected via resistors to a positive power supply voltage. This means that when the bus is free, both lines are high. All devices on the bus must have open-collector or open-drain pins. Activating the line means pulling it down (wired AND). The number of the devices on a single bus is almost unlimited – the only requirement is that the bus capacitance does not exceed 400 pF. Because logical 1 level depends on the supply voltage, there is no standard bus voltage.

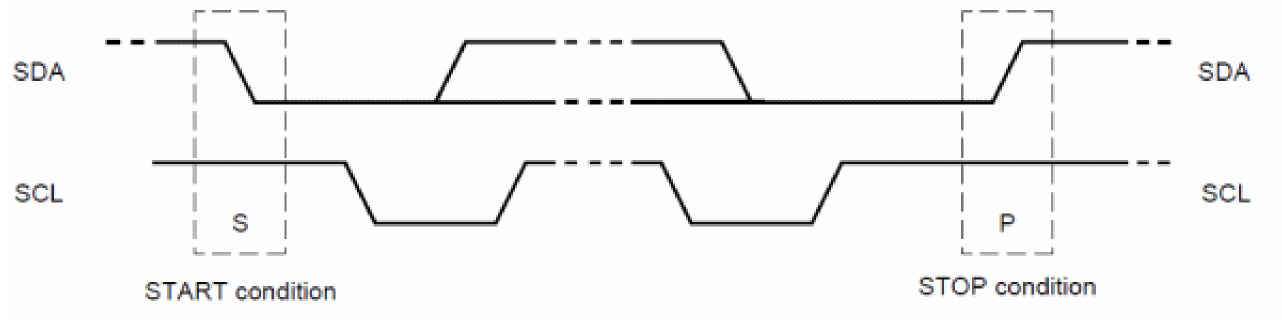
Serial Data Transfer

For each clock pulse one bit of data is transferred. The SDA signal can only change when the SCL signal is low – when the clock is high the data should be stable.



Start and Stop Condition

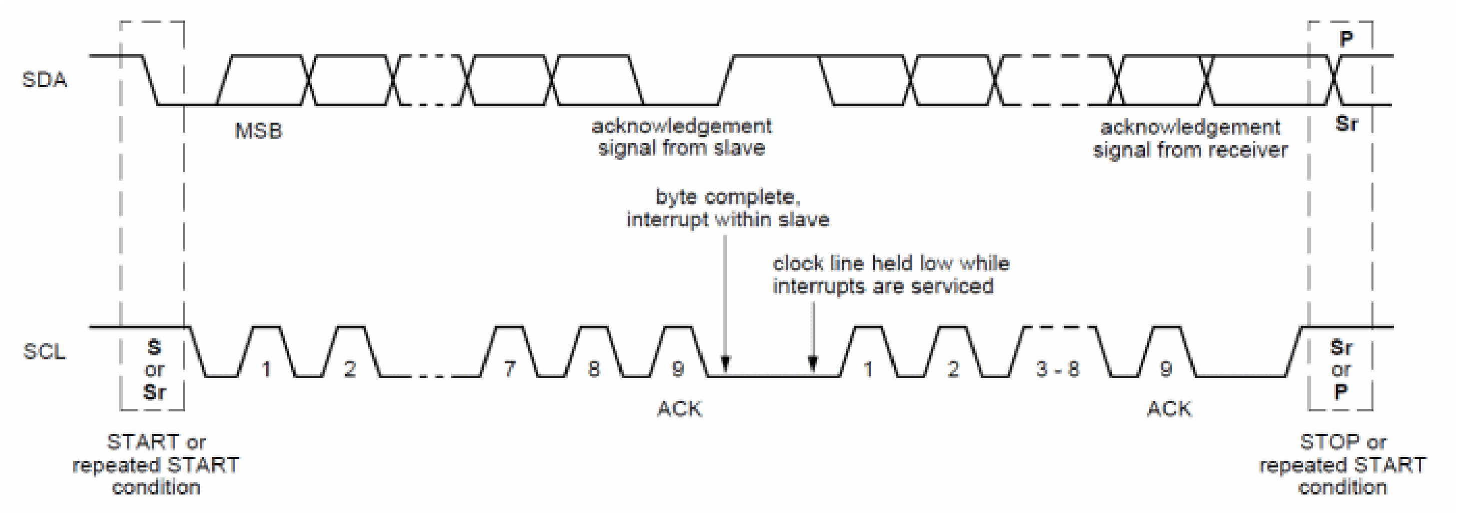
Each I2C command initiated by master device starts with a **START condition** and ends with a **STOP condition**. For both conditions SCL has to be high. A high to low transition of SDA is considered as **START** and a low to high transition as **STOP**.



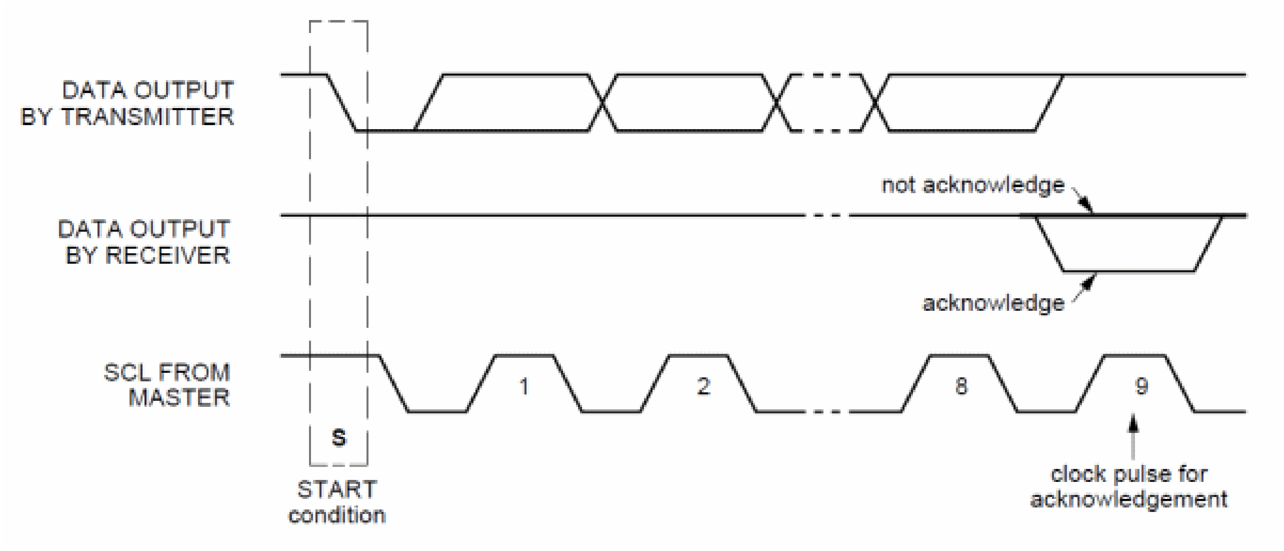
After the Start condition the bus is considered as busy and can be used by another master only after a Stop condition is detected. After the Start condition the master can generate a repeated Start. This is equivalent to a normal Start and is usually followed by the slave I2C address.

Microcontrollers that have dedicated I2C hardware can easily detect bus changes and behave also as I2C slave devices. However, if the I2C communication is implemented in software, the bus signals must be sampled at least two times per clock cycle in order to detect necessary changes.

I2C Data Transfer


SDA
SCL
MSB
acknowledgement signal from slave
byte complete, interrupt within slave
clock line held low while interrupts are serviced
acknowledgement signal from receiver
P
Sr
S or Sr
1
2
7
8
9
ACK
1
2
3 - 8
9
ACK
Sr or P
START or repeated START condition
STOP or repeated START condition

Data on the I2C bus is transferred in 8-bit packets (bytes). There is no limitation on the number of bytes, however, each byte must be followed by an Acknowledge bit. This bit signals whether the device is ready to proceed with the next byte. For all data bits including the Acknowledge bit, the master must generate clock pulses. If the slave device does not acknowledges transfer this means that there is no more data or the device is not ready for the transfer yet. The master device must either generate Stop or Repeated Start condition.



Synchronization

Each master must generate its own clock signal and the data can change only when the clock is low. For successful bus arbitration a synchronized clock is needed. Once a master pulls the clock low it stays low until all masters put the clock into high state. Similarly, the clock is in the high state until the first master pulls it low. This way by observing the SCL signal, master devices can synchronize their clocks.

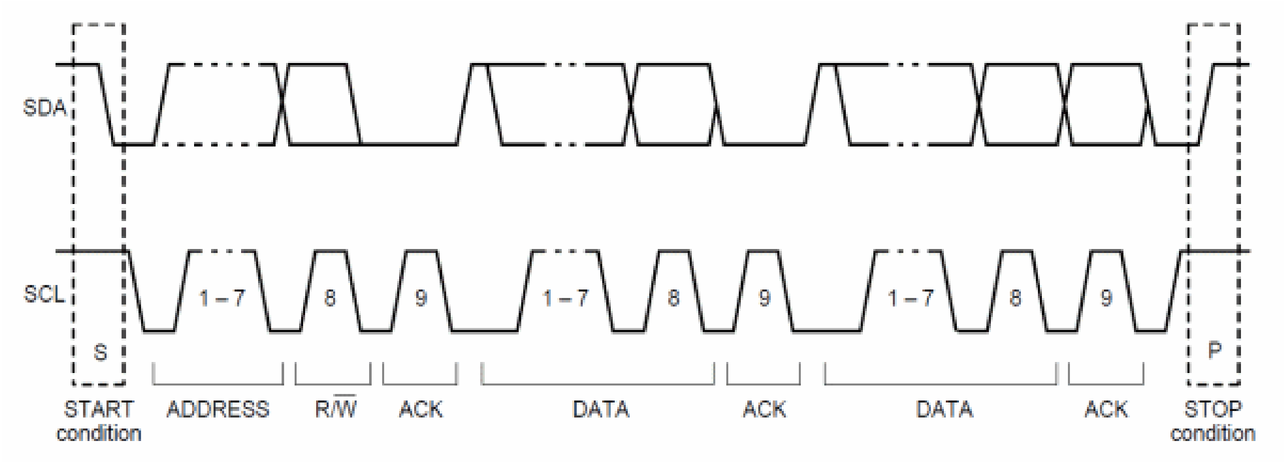
Arbitration

For normal data transfer on the I2C bus only one master can be active. If for some reason two masters initiate I2C command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. Master I2C device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each I2C master must monitor the I2C bus for collisions and act accordingly.

Clock Synchronization and Handshaking

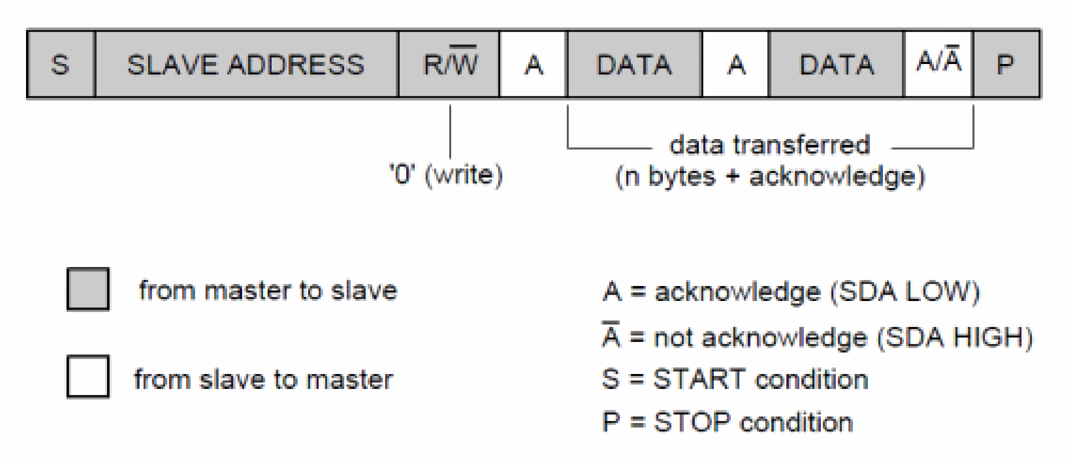
Slave devices that need some time to process received byte or are not ready yet to send the next byte, can pull the clock low to signal to the master that it should wait. Once the clock is released the master can proceed with the next byte.

Communication With 7-bit I2C Addresses

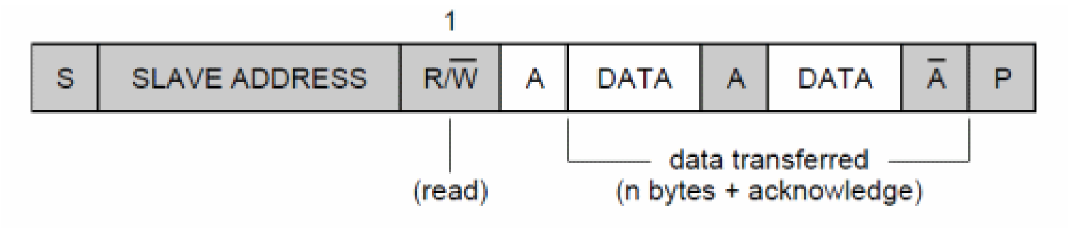


Each slave device on the bus should have a unique 7-bit address. The communication starts with the Start condition, followed by the 7-bit slave address and the data direction bit. If this bit is 0 then the master will write to the slave device. Otherwise, if the data direction bit is 1, the master will read from slave device. After the slave address and the data direction is sent, the master can continue with reading or writing. The communication is ended with the Stop condition which also signals that the I2C bus is free. If the master needs to communicate with other slaves it can generate a repeated start with another slave address without generation Stop condition. All the bytes are transferred with the MSB bit shifted first.

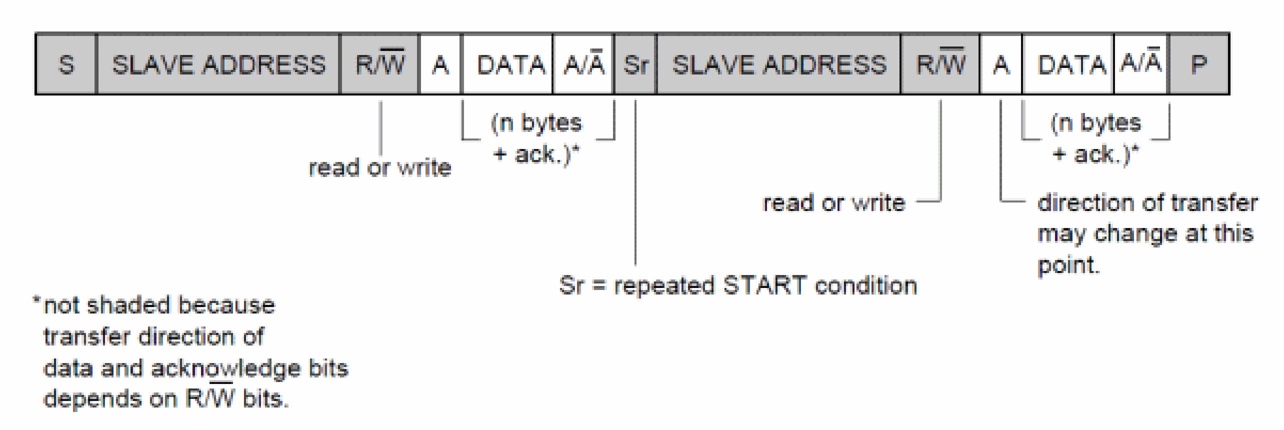
If the master only writes to the slave device then the data transfer direction is not changed.



If the master only needs to read from the slave device then it simply sends the I2C address with the R/W bit set to read. After this the master device starts reading the data.

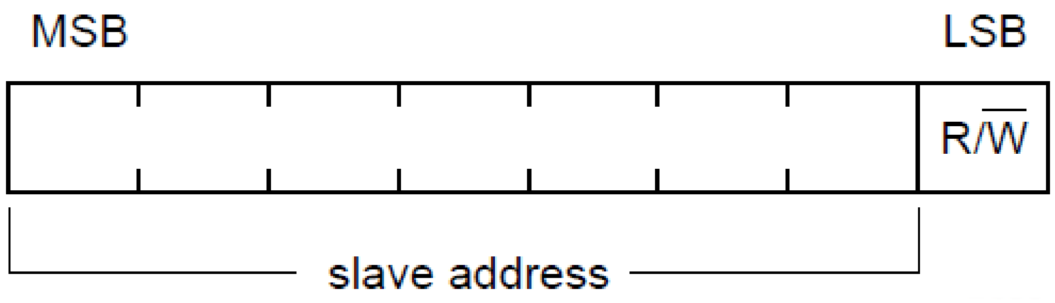


Sometimes the master needs to write some data and then read from the slave device. In such cases it must first write to the slave device, change the data transfer direction and then read the device. This means sending the I2C address with the R/W bit set to write and then sending some additional data like register address. After writing is finished the master device generates repeated start condition and sends the I2C address with the R/W bit set to read. After this the data transfer direction is changed and the master device starts reading the data.

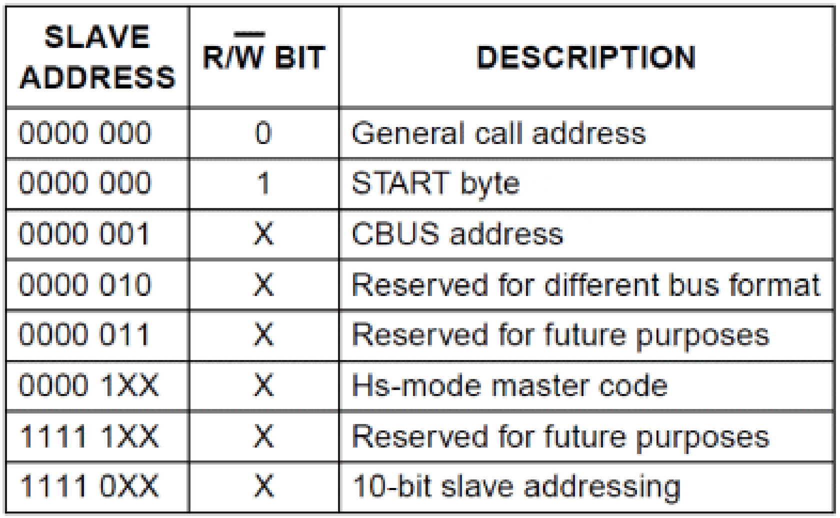

S
SLAVE ADDRESS
R/W̄
A
DATA
A/Ā
Sr
SLAVE ADDRESS
R/W̄
A
DATA
A/Ā
P
read or write
(n bytes + ack.)*
read or write
(n bytes + ack.)*
direction of transfer may change at this point.
Sr = repeated START condition
*not shaded because transfer direction of data and acknowledge bits depends on R/W̄ bits.

7-bit I2C Addressing

A slave address may contain a fixed and a programmable part. Some slave devices have few bits of the I2C address dependent on the level of address pins. This way it is possible to have on the same I2C bus more than one I2C device with the same fixed part of I2C address.



The allocation of I2C addresses is administered by the I2C bus committee which takes care for the allocations. Two groups of 8 I2C addresses are reserved for future uses and one address is used for 10-bit I2C addressing.

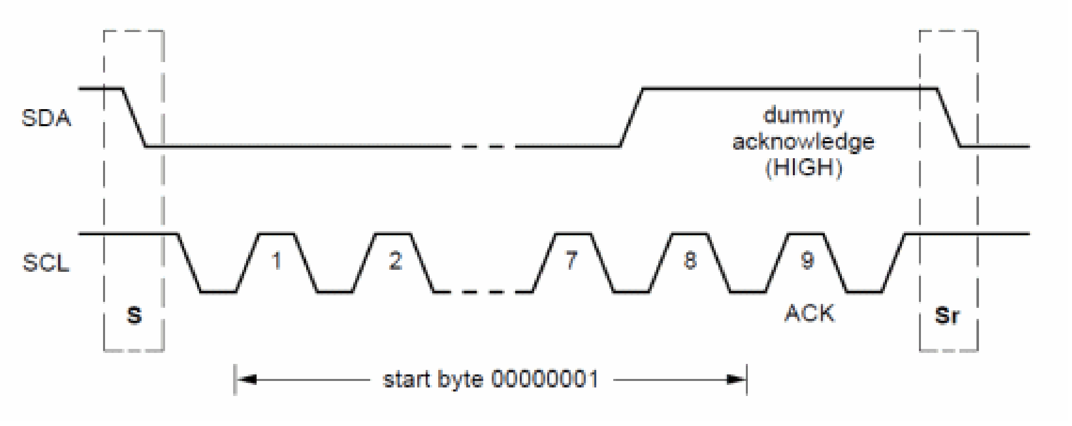


SLAVE ADDRESS	R/$\overline{W}$ BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

The general call address is used to address all devices on the slave bus. If any slave device doesn't need to respond to such call or general call is not supported by the slave device, the call must be ignored. If the device supports general call and wants to receive the data it must acknowledge the address and read the data as a slave receiver.

Start Byte

If microcontroller has I2C hardware and the microcontroller acts as a slave then the software needs to do nothing to check the bus state. The I2C hardware will detect Start condition, receive the I2C address and interrupt the software if necessary. However, if the I2C interface is implemented by the software, the microcontroller has to sample SDA line at least twice per clock pulse in order to detect changes. To simplify detection of I2C commands on the bus in such cases, a special I2C address called Start byte is used. Such start byte (0000 0001) is followed by an acknowledge pulse (for interface compatibility reasons). This combination holds the SDA line low for 7 clock pulses and allows simple detection of active I2C bus with lower sampling frequency.



Extension of the I2C Specifications

Standard mode of I2C bus uses transfer rates up to 100 kbit/s and 7-bit addressing. Such I2C interface is used by many hundred I2C-compatible devices from many manufacturers since its introduction in the 80s. However, with the advance of the technology, needs for higher transfer rates and larger address space emerged. There are cases where large amount of data needs to be transferred. Many complex embedded boards contain a large number of different I2C devices. In some cases it is very hard to avoid address collisions since 7 bits for I2C addresses allow only 127 different addresses where only 112 can actually be used. Some I2C devices on the board, despite address pins, have the same address. This resulted in few upgrades to the standard-mode I2C specifications:

- Fast Mode – supports transfer rates up to 400 kbit/s
- High-speed mode (Hs-mode) – supports transfer rates up to 3.4 Mbit/s
- 10-bit addressing – supports up to 1024 I2C addresses

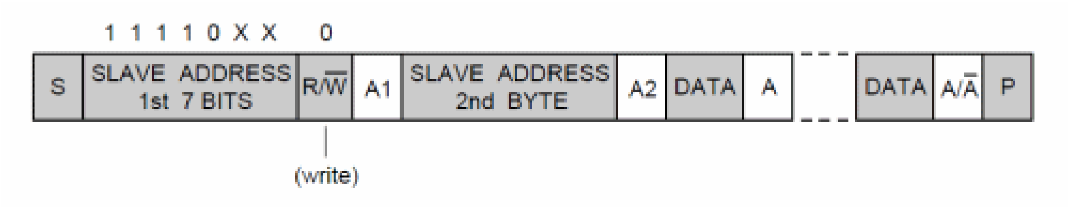
There can by any combination of the devices on the bus regardless of the supported speed and addressing. Fast mode devices are downward-compatible and can work with slower I2C controllers. However, most modern I2C controllers support all speeds and addressing modes.

High-speed mode uses signals called SCLH and SDAH to emphasize the higher speed. These signals are usually separated from standard SDA and SCL lines. High-speed mode introduces also few differences (or improvements) in the specifications:

- Improved data and clock line output drivers
- Schmitt trigger and spike suppression circuits on data and clock inputs
- Clock synchronization and arbitration is not used
- Clock signal has 1 to 2 high/low ratio

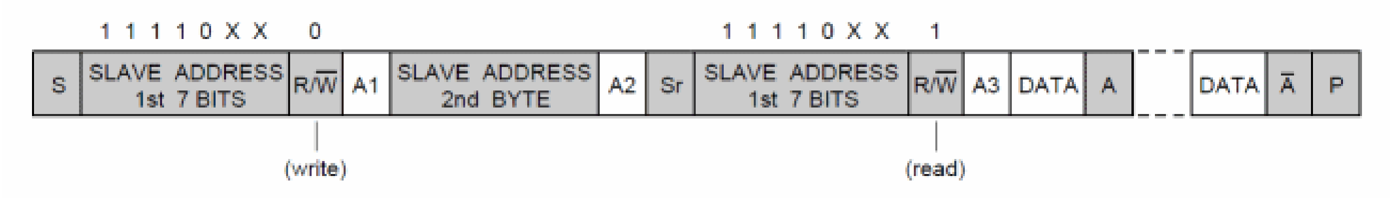
10-bit I2C Addressing

10-bit addressing can be used together with 7-bit addressing since a special 7-bit address (1111 0XX) is used to signal 10-bit I2C address. When a master wants to address a slave device using 10-bit addressing, it generates a start condition, then it sends 5 bits signaling 10-bit addressing (1111 0), followed by the first two bits of the I2C address and then the standard read/write bit.



If the master will write data to the slave device it must send the remaining 8 bits of slave address as the second byte.

If the master will read data from the slave device it must send the complete 10-bit address (two bytes) as for writing, then a repeated start is sent followed by the first address byte with read/write bit set to high to signal reading. After this procedure the data can be read from the slave device.



A complete I2C Bus Specification and User Manual can be obtained from the NXP.

Bus Pirate

A simple troubleshooting tool that communicates between a PC and any embedded device over most standard serial protocols, which include I2C, SPI, and UART- all at voltages from 0 - 5.5 V DC.

USB Logic Analyzer

Analyze up to 8 digital waveforms and timing information on your I2C, SPI, serial, and digital IO lines.

16-Channel USB Logic Analyzer

Analyze up to 16 digital waveforms. Automatically decodes the SPI, I2C, Serial, CAN, 1-Wire, I2S, PCB, UNI/O, and Manchester protocols.